VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2014 **Zdeněk Balicki**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Hra Bang! v prostředí internetu
Game Bang! Over Internet

2014 **Zdeněk Balicki**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Zdeněk Balicki**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Hra Bang! v prostředí internetu
Game Bang! Over Internet

Zásady pro vypracování:

Cílem práce je vytvořit hru Bang! v prostředí internetu v jazyce java, která umožní před začátkem hry volit jednotlivá rozšíření hry.

Hra bude umožňovat:
1. Možnost volby rozšíření hry HighNoon nebo DodgeCity.
2. Hra proti jednoduché umělé inteligenci.
3. Hru více hráčů přes internet.
4. Hra se bude spouštět prostřednictvím technologie Java WebStart.
5. Program umožní připojení do současně vyvíjeného portálů her v jazyce Java.

Práce bude obsahovat:
1. Implementaci hry Obrana věže.
2. Programátorskou dokumentaci řešení s využitím diagramů jazyka UML.
3. Uživatelskou dokumentaci aplikace.

Seznam doporučené odborné literatury:

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025
[2] DARWIN, Ian F. Java cookbook. 2nd ed. Sebastopol, CA: O'Reilly, c2004, xxiv, 829 p. ISBN 05-960-0701-9. Dostupné z: http://it-ebooks.info/book/2249/
Dále podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 7. května 2014 

……

podpis

Poděkování

Rád bych poděkoval **Ing. Davidu Ježkovi, Ph.D.** za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Dále bych chtěl poděkovat všem, kteří mě v těch nejtemnějších chvílích nerušili telefonáty, IM zprávami nebo vrtačkami.

Abstrakt

Hlavním účelem mé bakalářské práce je vytvoření implementace westernové karetní hry Bang! v jazyce Java. Program slouží ke hraní s dalšími hráči přes internet a umožňuje doplnění nedostatků hráčů primitivní umělou inteligencí. Důležitou součástí základní hry jsou i její rozšíření s názvy Dodge City a High Noon, které mění chod hry a přidávají další karty různých druhů. Tato rozšíření lze podle volby kombinovat při vytváření nové hry. Program je spustitelný prostřednictvím známé technologie Java Web Start, aby mohl být spolu s dalšími aplikacemi elementem herního portálu. Implementaci je potřeba správně popsat z programátorského i uživatelského hlediska a znázornit řádnými UML diagramy a dalšími schématy pro vysvětlení funkcí programu. Toho se týká tento dokument.

Klíčová slova

Java Web Start, Bang!, Město duchů, Pravé poledne, karetní hra, online více hráčů

Abstract

The main purpose of my bachelor thesis is to create in the Java language an implementation of the wild west-themed card game called Bang! The program is for multiplayer gaming over internet and allows the possibility of adding primitive AI players. An important part of the basic game are its expansions named Dodge City and High Noon that change the gameplay and add new card varieties. These expansions are combinable during creation of a game. The program is Java Web Start executable, in order to be an element of a game portal, along with other Java applications. It is necessary to correctly describe the implementation by technical and user perspectives including proper UML diagrams and other schemes clarifying functions of the program. That is what is written in this document.

Key words

Java Web Start, Bang!, Dodge City, High Noon, card game, online multiplayer

Seznam použitých symbolů a zkratek

.jar	Java Archive - formát souboru platformy Java
.jnlp	Java Network Launching Protocol - formát souboru Javy Web Start
.pdf	Portable Document Format - formát přenositelných dokumentů
.png	Portable Network Graphics - formát souboru pro bezeztrátovou kompresi rastrové grafiky
AI	artificial intelligence (umělá inteligence)
AWT	Abstract Window Toolkit - knihovna základních grafických prvků
B	byte (bajt)
CSS	Cascading Style Sheets - popis zobrazení www stránek
DC	Dodge City (Město duchů)
HN	High Noon (Pravé poledne)
HTML	HyperText Markup Language - jazyk pro vytváření www stránek
IDE	Integrated Development Environment - vývojové prostředí
IM	instant messaging
JDK	Java Development Kit - produkt firmy Oracle obsahující nástroje pro vývoj aplikací
JRE	Java Runtime Environment - rozhraní ke spouštění Java aplikací
KB	kilobyte (kilobajt)
MB	megabyte (megabajt)
UML	Unified Modeling Language - grafický jazyk pro modelování aplikací
URL	Uniform Resource Locator - specifikace umístění zdrojů informací
XML	Extensible Markup Language - obecný značkovací jazyk

Obsah

1. Úvodem

V tomto dokumentu je popsána mnou naprogramovaná implementace karetní hry Bang!, jejíž autorem je Ital *Emiliano Sciarra*. Text je rozdělen především do tří kapitol. První zkoumá herní možnosti, jejich mechaniku a problematiku, představuje bázi pravidel. Druhá detailně popisuje hru z pohledu programování (programátorská dokumentace), člení logiku aplikace, věnuje se použitým technologiím a poskytuje seznam tříd a dalších součástí, jejichž vztahy jsou znázorněny přehlednými diagramy. Třetí kapitola se zabývá vysvětlením aplikace běžnému uživateli (uživatelská dokumentace), nahlíží na to, co de facto bylo programováno.

Příležitost zrození internetové hry mě zaujala, proto jsem si toto téma bakalářské práce zvolil. Hra Bang!, publikována v roce 2002, se stala vhodným adeptem pro možnost vyzkoušet a pochopit rozšiřování hry novými doplňky. Zároveň mě oslovila i myšlenka vedoucí k úvaze, jak by vůbec aplikace měla fungovat, vypadat a být ovládána. Na základě toho vznikla solidní výzva. Začal jsem zjišťovat, zda se už někdo o něco podobného někdy pokoušel. Výsledkem pátrání byla víceméně jen jedna neoficiální verze umožňující hraní pouze základního Bangu! Programována byla v jiném jazyce a její kódy musely být staženy z internetu. Z toho se stal sen s cílem vytvoření bezplatné aplikace na vysoké úrovni, ten se ale krátce nato rozplynul, jak je později popsáno.

Již zpočátku jsem tušil, že oproti hraní her nebude jejich vytváření žádná hračka. Bylo potřeba prostudovat dříve mi zčásti známá a přesto komplikovaná pravidla a najít jejich společné prvky, podle kterých by se hierarchie rozhraní a tříd mohla odvíjet. Sdílených vlastností jsem však moc neobjevil a začal si jich všímat spíše až při samotném programování.

Proto kontrast očekávání a postupných poznatků budu zmiňovat v obou dokumentacích, spolu s novými nápady vzniklými na jejich základě. V programátorské (Kapitola 3) rozebíráním samotných základů dědění a průběhů komunikace přes internet, zatímco v uživatelské (Kapitola 4) posuzováním možných negativních rysů funkcí a vzhledu. Vlivem subjektivního pohledu na mé dílo je totiž zaručen nesouhlas s vývojem aplikace a způsoby prezentování stavu hry těm, kteří ji mají hrát. Nedocílil jsem tak zdaleka posledního stádia jejího vývoje.

Díky testování jsem dokázal opravit majoritní chyby, které kolikrát vznikly třeba jen naprosto banálním způsobem. Neopravené chyby popsané v dokumentacích by tak byly po nasazení aplikace známy, takže při zkušebním provozu by existovalo více prostoru pro nacházení nových závad.

2. Analýza hry

2.1 Popis Bangu!

Hra je v základní verzi pro 4-7 hráčů, po přidání rozšíření Dodge City až 3-8 hráčů se speciálními pravidly. Uprostřed hrací plochy je lízací a odhazovací balíček. Rozšíření High Noon, případně neimplementované A Fistful of Cards dále nabízí speciální balíčky s kartami odlišnými od hracích.

Každému hráči je před začátkem Bangu! vylosována karta role, ten ji zná, ale ostatním hráčům je utajena. Jediná veřejná role je role šerifa, proto ji obrátí lícem nahoru. Role znamenají poměr dobra (šerif, pomocníci šerifa) a zla (bandité), odpadlík je neutrální - bojuje proti silnější straně sám za sebe a snaží se zůstat ve hře jako poslední naživu. Zároveň každý hráč dostane náhodnou kartu postavy, která mu stanoví maximální počet životů a přidává zvláštní vlastnosti, popř. schopnosti. Všechny postavy jsou všem známy. Hrací karty má hráč v ruce nebo před sebou na stole.

Počínaje šerifem, hráči se střídají ve směru hodinových ručiček. Tah hráče se skládá ze tří fází - líznutí dvou karet, zahrání hracích karet, odhození karet (tak, aby hráč měl v ruce maximálně tolik karet, kolik má aktuálně životů).

Postavám (hráčům) jsou během hraní ubírány a přidávány životy. Pokud postava nemá způsob, jak se zachránit, umírá a je vyřazena ze hry. Při zapnutém rozšíření High Noon je však vhodné hru dále sledovat, neboť se mrtvé postavy na jedno kolo do hry vrací.

Konec hry nastává, pokud je šerif zabit, nebo pokud jsou zabiti všichni bandité a odpadlík (odpadlíci). Zvláštní případ je u speciální hry 3 hráčů (rozšíření Dodge City), ve které vůbec není role šerifa.

Toto je obecný popis průběhu hry, nenahrazuje pravidla Bangu! [1].

2.2 Karty

Karty jsem tedy rozdělil na karty rolí, postav, hrací a speciální. Hrací karty jsem dále rozdělil na obyčejné a stolní - s modrým okrajem a zeleným okrajem (Dodge City). Všechny hrací karty mají svou hodnotu a barvu, stejně jako nejběžnější druh karet, avšak oproti nim jsou specifické i svým *názvem* a *efektem*.

Příklad: Karta s názvem Bang! má efekt Bang!, karta s jiným názvem může mít efekt Bang!, ale nepovažuje se za kartu Bang! Toto slovíčkaření je důležité hlavně v souvislosti s FAQ autorovi hry.

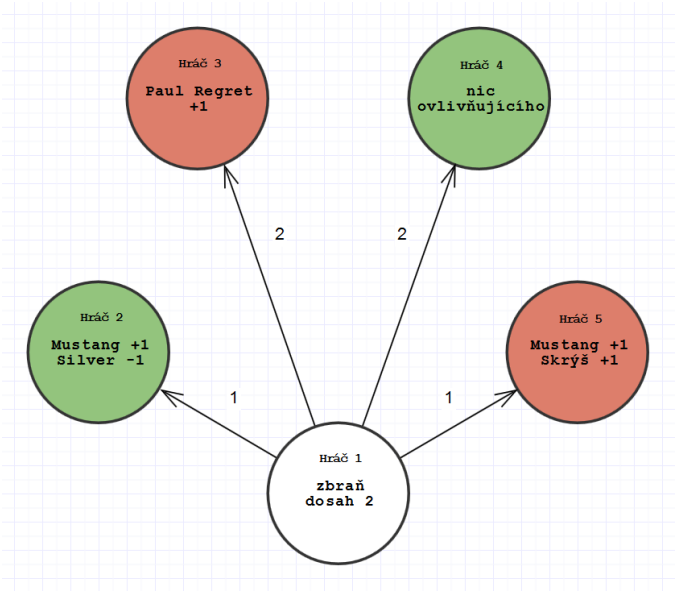
2.2.1 Aktivní ability

Symboly na kartách udávají kromě efektu i další upřesnění (především cíl efektu *target*), proto se vše oplatilo brát jako jeden celek, který jsem začal nazývat *abilita* (schopnost). Základní operace s kartami *Lízni*, *Odhoď* a *Dej na stůl* jsem začal vnímat rovněž jako efekty a stejně tak jsou i obaleny abilitou. Všechny typy abilit - efekty mohou být použity jen v daných fázích tahu.

2.2.2 Pasivní ability

K abilitám, a tedy aktivním schopnostem, jsem přidal i vlastnosti - pasivní ability, které se týkají modrých stolních karet a vlastností postav. Jednou z pasivních abilit je lízání, které ovlivňují postavy. Výchozí počet jsou 2 karty z balíčku, jenže počet i způsoby se můžou měnit - jako například *Bill Noface*, který si líže 1 kartu a plus jednu za každý ztracený život.

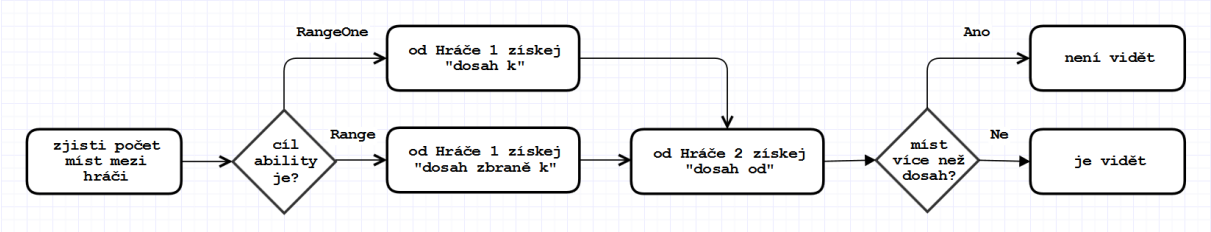
Podstatnější pasivní abilita je ta ovlivňující vzdálenost mezi hráči. Výchozí dosah jednoho živého hráče na druhého živého hráče vedle něj je 1, na hráče ob něj 2 atd. Vzdálenost platí oběma směry hodinových ručiček.



Obrázek 1 - Vzdálenost mezi hráči

Dosah záleží i na používané abilitě - zda musí být cíl ve vzdálenosti jedna nebo ve vzdálenosti zbraně. Při první možnosti by vliv měly jen modré karty vyložené před hráčem, případně vlastnost jeho postavy (*Paul Regret*, *Rose Doolan*). Pokud by se však jednalo o cíl v dosahu zbraně, do výpočtu by musela zasahovat její hodnota, jak je vysvětleno na diagramu [Obrázek 1], které se týká revolveru *Schofield* s dosahem 2. Hráč 2 je označen zeleně, protože na něj lze zbraní dosáhnout, karta *Silver* by ovlivnila hodnotu jen tehdy, pokud by abilitu použil on.

Proto jsem třídě předpisující hráče vytvořil metody pro výpočet hodnot od a směrem k ostatním. Instance hry pak na základě těchto hodnot a abilitě určí, jestli hráč dosáhne.



Obrázek 2 - Zjednodušené znázornění kontroly dosahu

K algoritmu pro výpočet viditelnosti [Obrázek 2] byly později doplněny další cíle abilit, upřesnění podmínek hráčů a ošetření parametrů pro obecnější používání. Metoda takto může například kontrolovat i karty s účinkem na hráče kartu používající (cíl ability *Self*).

2.2.3 Efekty

Protože se implementace fyzické karetní hry z pohledu logiky lehce liší, navrhl jsem vhodnější způsob definování efektů karet.

Draw (Lízni), Discard (Odhoď), ToTable (Dej na stůl)
Nejsou přímo efekty karet určené symboly, ale základní operace s kartami.

Loot (Získej kartu), Take (Vezmi kartu)
V praxi jsou totéž jako *Líznout*, jenže se používají v jiné fázi tahu a každá má jiný zdroj - balíček nebo karty jiného hráče.

Destroy (Znič kartu)
Odhodí kartu jinému hráči, logicky a fázově odlišné od *Odhoď*.

Attack (Bang!), Missed (Vedle!), Regenerate (Pivo)
Hlavní efekty podle symbolů herních karet.

2.2.4 Cíle

Self (Na sebe)
Karta bez symbolu s cílem efektu na sebe.

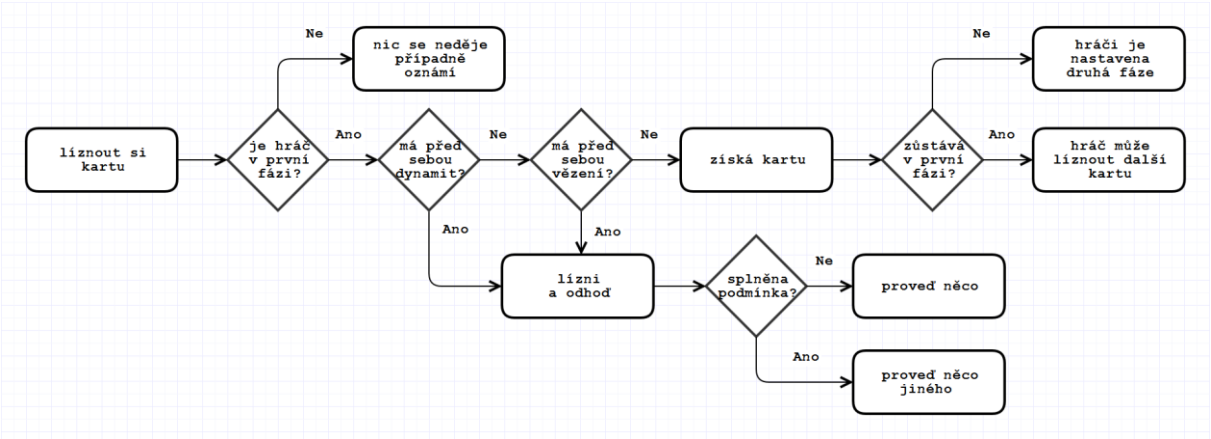
RangeOne (Dosah jedna), Range (Dosah zbraně), Any (Libovolný hráč), Others (Všichni ostatní)
Cíle podle klasických symbolů.

All
Karta má efekt na všechny hráče včetně hráče, co ji použil. Symbol na kartě je zapsán složitěji.

2.3 Tahy hráčů

Bylo potřeba zavést více fází než pouze *první*, *druhou* a *třetí*. Při inicializaci hry a mimo svůj tah jsou všichni hráči *neaktivní* (*Inactive*). Tah hráče začíná první fází, která se po potřebném počtu líznutí sama změní do fáze druhé. Pokud má hráč před sebou karty jako jsou *Dynamit* nebo *Vězení*, dochází k automatickému odhození líznutých karet (*Lízni a odhoď*), takže jejich kontrola před první fází nemusí být.

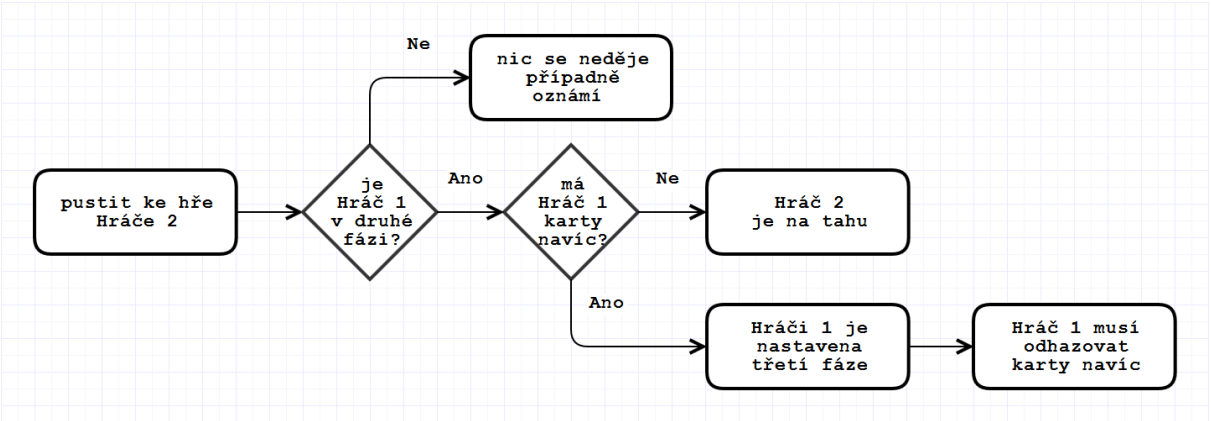
Schéma [Obrázek 3] představuje princip první fáze hráčova tahu a vliv karet se speciálním efektem:



Obrázek 3 - Zjednodušené znázornění lízání karet

Jakmile se hráč rozhodne (nebo nemá na výběr) pustit k tahu dalšího hráče a musí odhazovat přebývající karty, zůstane ve třetí fázi, dokud neprovede poslední nutné odhození. Jestli nemá žádné karty navíc a chce ukončit druhou fázi, stává se přímo neaktivním a na řadě je další hráč s fází jedna.

Další schéma [Obrázek 4] znázorňuje případ, kdy hráč chce ukončit svou druhou fázi:



Obrázek 4 - Zjednodušené znázornění konce hráčova tahu

Když hráč na někoho útočí, je v pomocné neaktivní fázi *zahálející* (*Idle*). Hráč reagující na útok je v pasivní *obranné* fázi. Po obraně se zahálející hráč vrací zpět do druhé fáze. Podobné situace se týkají dalších pasivních fází, kdy je mimo svůj tah potřeba něco vybírat.

V případě, že abilita karty má jako cíl útoku jednoho hráče (dosah jedna, dosah zbraně, libovolný hráč), situace je bezproblémová. Avšak při hromadném útoku na všechny ostatní hráče bylo žádané dodržet pořadí jejich reakcí k obraně. Více je popsáno v kapitole 3.2.

2.4 Interakce

Každý hráč ví, v jaké fázi je a v jaké jsou ostatní hráči, proto nebylo nutné na změnu fáze zvlášť upozorňovat, o to se stará zobrazení hry a její ovládání. Zato pro umělou inteligenci musel být vymyšlen způsob, jak má hra potřebu reakce umělým hráčům oznamovat. Konkrétně je toto vysvětleno u popisu implementace.

3. Programátorská dokumentace

Program je napsán v jazyce Java (J2SE) za podpory JDK 1.7, místy je použito jednoduché HTML/CSS a XML (Java Web Start). Jeho hlavní struktura se dá především rozdělit na 3 části: modelová, komunikační a prezentační. Jak je dále popsáno, i po vyzkoušení několika samostatných knihoven nebylo potřeba některé z nich využít. U prezentační části byl kladen velký důraz na práci s knihovnou Swing (a starší AWT), jinak je program vázán na běžné knihovny jako např. kolekce, eventy, serializace, streamy aj. Projekt je vytvořen v IDE Netbeans.

Nejzásadnější část (modelová) představuje stav hry, hráčů a karet a většinu jejich logiky. Komunikace probíhá podle architektury klient-server, kdy klienti vytvářejí hry na serverové části, která na základě požadavků rozesílá zpět výsledky všem klientům připojeným ke hře. Na obou stranách se vytvoří instance hry a server klientům oznamuje jednoduchými příkazy provedené změny, čímž jejich verze zachovává rovněž aktuálními. Třetí část programu potom prezentuje klientům stav hry a umožňuje jim hru ovládat a komunikovat s jinými klienty - hráči.

Tyto tři části jsou pak v jedenácti balících (Java packages) s větvením začínajícím na bang, connection a program. Ke každému tak připadá jedna z částí. V balíku program se nalézá .jnlp soubor technologie Java Web Start spolu se spustitelným programem klienta, speciální kostra programu ke spuštění serveru je v balíku s názvem *program.server*.

Všechny soubory tříd, rozhraní, popř. enumů jsou okomentovány stylem pro generování Javadocu. Jelikož je projekt celkem rozsáhlý - přes 13000 řádků kódu, komentování bylo časově náročné. Proto jsou popsány kromě souborů jen jejich konstruktory a metody public/protected, jež stály za zmínění. Občas jsou zdůrazněny důležité metody s úrovní přístupu private, nikoli však statické a jiné atributy. S lepší hierarchií rozhraní a vhodnou volbou specifikátorů přístupů by komentování bylo jistě snadnější, moudré je jistě i programování tříd s přiměřeným počtem metod. Velikost vytvořeného Javadocu popisující můj program je cca 4,05 MB.

Při psaní kódů jsem se snažil dodržovat používané Java konvence [2], ale pár způsobů formátování jsem mermomocí chtěl podle sebe. Patří mezi to především složené závorky na novém řádku, pojmenování neintegerových statických atributů jinak než Caps Lockem, obzvlášť když se týkaly tak složitých názvů karet, nepoužívání mezer mezi přepisovanými metodami listenerů, závorky některých try-catch bloků a šířka řádků 100 znaků, když už dnes převládají širokoúhlé monitory. Také atributy jsem se snažil sdružovat raději logicky než podle datových typů či klíčových slov. V případě týmového programování by nebyl problém styl formátování v jakémkoli IDE změnit.

Eventy a listenery (popř. testovací výpisy na konzoli) jsou v dokumentaci zmíněny minimálně z důvodu jednoznačnosti jejich významu. Podle názvů se totiž dají lehce pochopit.

3.1 Model hry

V této části programu bylo podstatné zamyslet se nad tím, jak má hra vůbec probíhat a jak se s kartami bude zacházet. Původní postup zahrnoval použití velkého množství enumů, to se ukázalo jako špatné řešení z důvodu nemožnosti je jakkoli rozvíjet a snaha je zkombinovat s rozhraními byla neefektivní a komplikovaná. Proto jsem se tomu snažil vyvarovat a v konečném důsledku jich lze najít jen pár. Zpětným pohledem na model při rozšiřování hry se začaly rýsovat představy o ještě optimálnějším způsobu dědičnosti, jenž by tento proces usnadnil. Některých nápadů jsem využil a strukturu vylepšil s touhou přiblížit se k ideálu, kterého by se nejspíše dosáhlo například pomocí více rozhraní.

balíky

bang	Obsahuje prvky spojené se základní hrou. Hlavními třídami jsou Bang a Player a rozhraní Expansion
bang.card	Definuje, aktivní a pasivní ability karet a jejich další vlastnosti. Tvoří jejich sktrukturu.
bang.dodgecity	Rozšíření Město duchů.
bang.highnoon	Rozšíření Pravé poledne

k balíkům card, dodgecity a highnoon dále připadají složky /images s obrázky symbolů a nových karet

3.1.1 Základní hra

Model hry byl vytvářen tak, aby byl použitelný na serverové i klientské straně, protože klient nezná většinu hracích karet i rolí, má je *Unknown*. Přesto musí být na obou stranách dodržovány odpovídající počty karet a životů - na straně klienta se jen tak z prázdna nemůže objevit karta navíc.

bang.Bang
Hra se skládá z karetních balíčků a instancí hráčů. Řídí její základní průběh (směr hry, nové kolo, hráč na tahu a další) a interakce mezi hráči a kartami. Udává i výsledek hry. Metoda *addExpansion* přidá k základní hře rozšíření rozhraní Expansion. Metoda *setSlots* určuje počet hráčů a jejich pořadí podle náhodného rozdělení. Metoda *prepareCards* zamíchá karty podle nastavení. Metoda *assignIdentities* určí role a postavy. Metody *drawCard*, *takeCard*, *discardCard*, *drawAndDiscard* a *cardToOtherTable* provádí základní operace s kartami. Metoda *isVisible* vypočítává, zda jeden hráč dosáhne na druhého, to závisí na hráčových kartách, vlastnostech i na abilitě. Metoda *getBalanceTheory* odhaduje dominantní stranu a metody *getWinner* a *setWinner* stanovují vítěze.

bang.Expansion
Rozhraní rozšíření Bangu! Hra provádí metody rozhraní v potřebných chvílích a záleží na každém rozšíření, jak zareaguje - nemusí v dané chvíli dění nijak ovlivnit. Metody *changeMinPlayers* a *changeMaxPlayers* změní minimální a maximální počet možných hráčů, *getRoles*, *getCharacters*, *getPlaying* do hry přináší nové karty. Metody *prepareCards*, *newInTurn*, *newRound* jsou vykonávány při rozdávání karet, v okamžiku, kdy je další hráč na tahu, a na začátku nového kola.

bang.Player, bang.PlayerPerson, bang.PlayerAI
Abstraktní třída Player se dělí na skutečné a AI hráče. Toto dělení by šlo samozřejmě provést i jiným způsobem. Hráči mají jméno, určitý počet životů, karty na ruce a stole. Nastavuje se zde jejich fáze, vypočítává viditelnost od dalších hráčů, uchovává počet použitých akcí a pracuje s jejich kartami. Eventy nad ArrayListy jsou oznamovány způsobem vlastních metod. Zpětným pohledem by byla lepší varianta použití vlastních kolekcí.

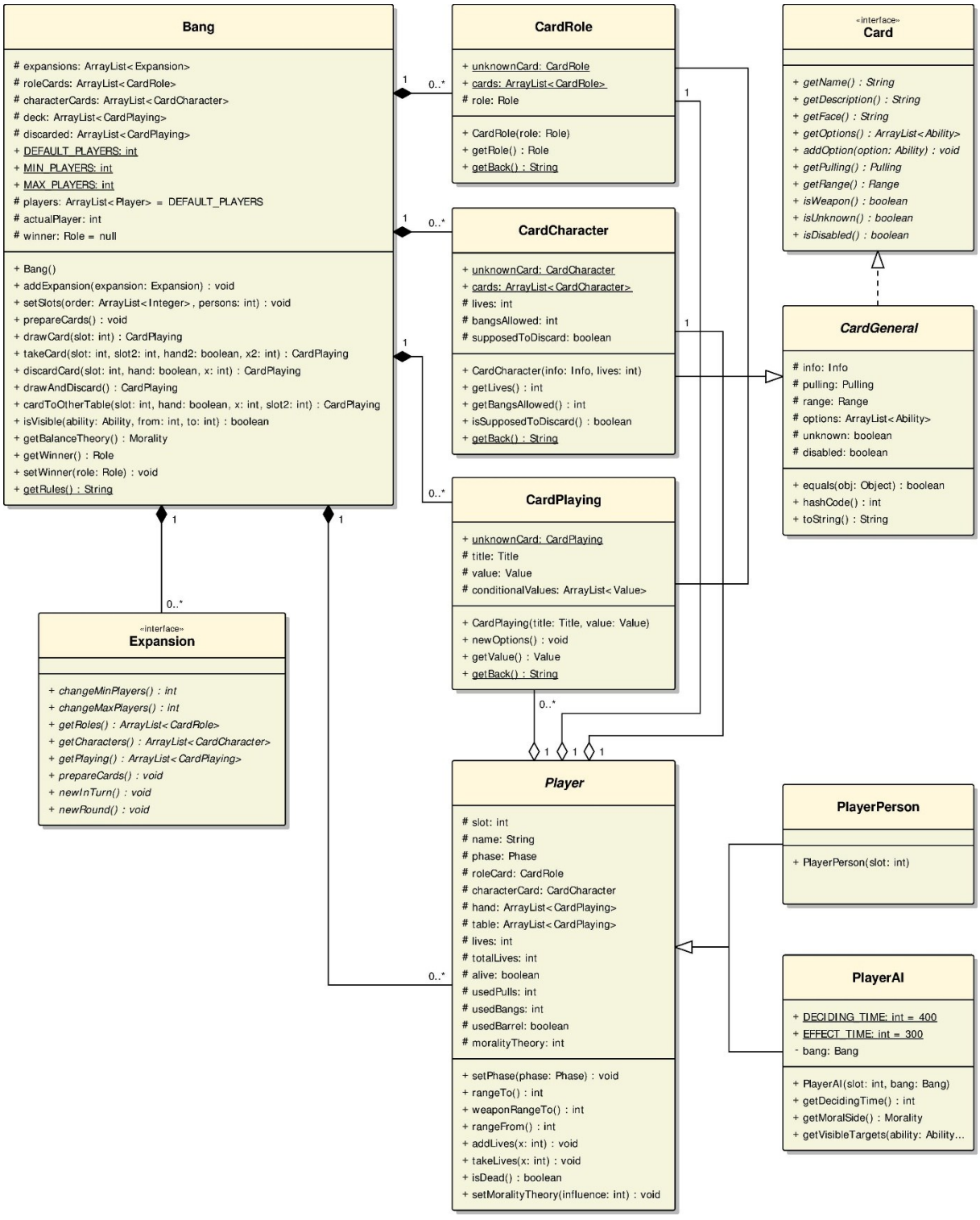
Metoda *setPhase* mění jejich fázi. Metody *rangeTo*, *weaponRangeTo*, *rangeFrom* vrací vliv vzdálenosti k ostatním hráčům a od nich, *addLives* a *takeLives* přidávají a odebírají životy. Další významné metody jsou pak *isAllowedToPull*, *isAllowedToUseBang*, *isAllowedToUseBarrel*, *isSupposedToDiscard*, zbytek pracuje s kartami hráčů. Za povšimnutí stojí metoda *setMoralityTheory*, která podle provedených akcí hráče určuje, na které morální straně pravděpodobně je.

bang.Phase
Enum fází hráčova tahu a těch mimo jeho tah.

bang.Target
Enum vybraného hráče nebo více cílů.

bang.Morality
Enum morálky - vztahu mezi dobrem a zlem.

Obrázek 5 - Diagram modelu základní hry



3.1.2 Struktura karet

card.Card, card.CardGeneral
Rozhraní všech karet a abstraktní třída CardGeneral implementující hlavní metody pro zjištění jména *getName*, popisu *getDescription*, cesty k obrázku karty *getFace* atd. Metoda *getBack* vracející cestu k rubové straně karty je statická, proto je u potomků definována podle potřeby.

CardRole, CardCharacter, CardPlaying, CardCommon, CardTable, CardBlue
Různé druhy karet - role, postavy, hrací karty. CardPlaying slouží k vytvoření neznámé hrací karty, Common a Blue jsou pro karty konkrétní. Třída CardTable vytváří spoj pro CardGreen. K tomu by ve skutečnosti došlo spíše za logického předpokladu, jak struktura karet bude dále v rozšířeních ovlivněna. Pohled na způsob dělení hry před a po implementaci rozšíření se tak stal zásadní, ale některé části jsou přesto vytvořeny pro usnadnění tak, jako by se předem znal návrh toho, co hra v budoucnu bude umět. U modelu by se všeobecně mělo počítat s dalšími možnými rozšířeními, i když není ještě známo, jak budou vypadat. V ideálním případě by tak dříve vytvořené soubory kódu vůbec nemusely být změněny. Metoda *newOptions* pro všechny karty dědící z CardPlaying nastavuje povolené akce.

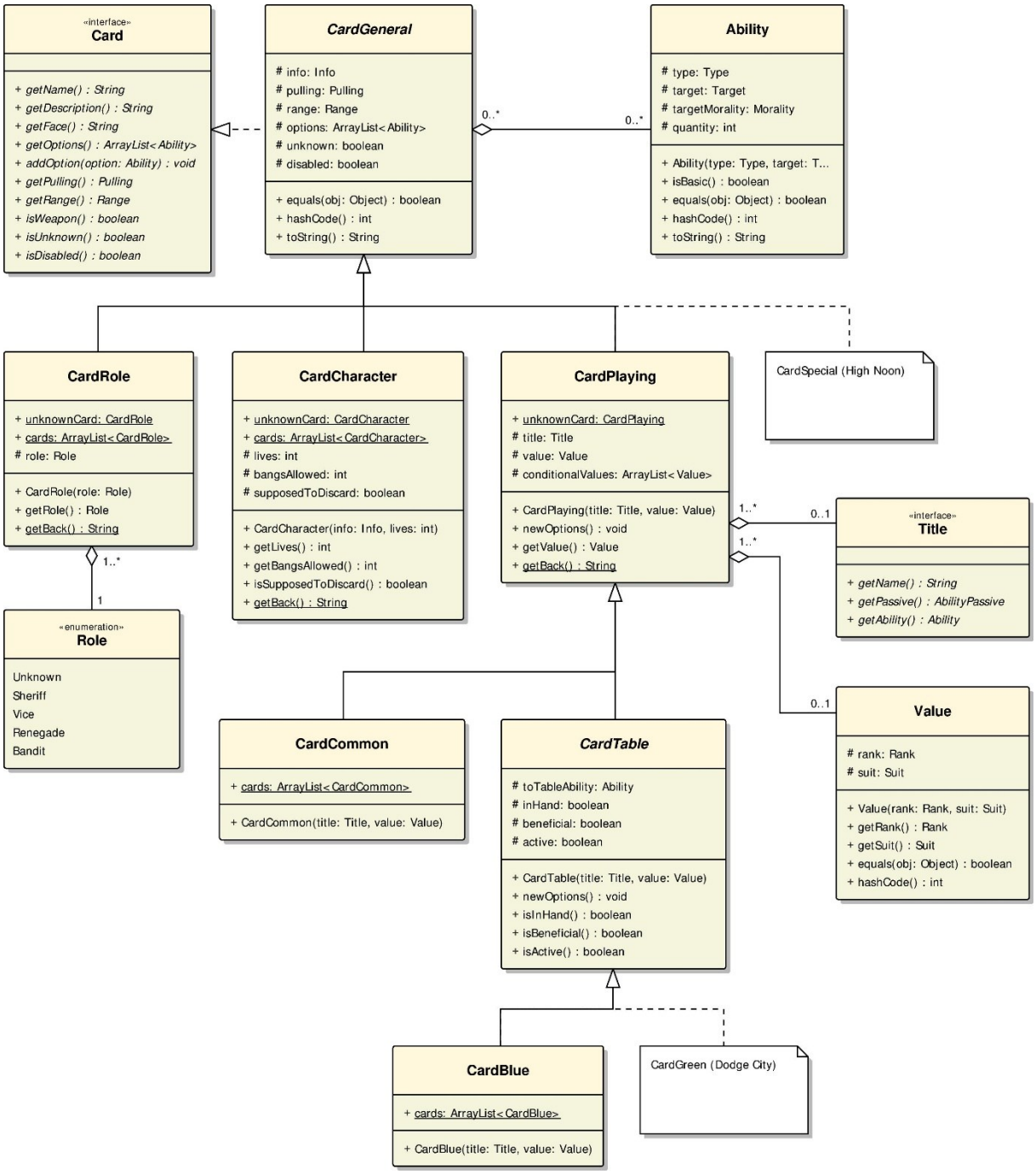
card.Ability, card.AbilityPassive
Ability jsem rozdělil na aktivní a pasivní. Aktivní se týkají karet hracích, pasivní i karet postav. Zatímco pasivní pouze mění některé vlastnosti, aktivních abilit je několik typů použitelných v různých fázích tahu. Zároveň souvisí s různými cíli a jako je tomu u karty *Vězení*, může záležet na tom, zda je cílem ability šerif, proto je potřeba uvést i morálku cíle. Posledním údajem aktivních abilit je počet (síla), kolikrát má být použita. Typickými příklady pasivních abilit je Range ovlivňující vzdálenost mezi hráči a Pulling ovlivňující lízání karet v první fázi tahu.

card.Info, card.Value
Třída Info definuje název a popis karet, třída Value hodnotu karty Rank a její barvu Suit.

card.Title, card.TitleBang
Hrací karty stejného názvu se liší jen hodnotou a barvou, proto bylo potřebné zavést část, která je spojuje. I v logice hry se karty dělí na karty „s názvem“ a karty „s efektem“. Title je rozhraní a TitleBang konkrétní třída s titulem a aktivní nebo pasivní abilitou.

card.CardException
Dědí z třídy Exception a může nastat při operacích s kartami. Většinou neovlivní průběh hry a vypíše se na konzoli z informativního důvodu.

Obrázek 6 - Diagram struktury karet



3.1.3 Rozšíření

Logika rozšiřování hry mohla být provedena několika způsoby, proto jsem od každého vyzkoušel něco a zjistil tak výhody i nevýhody. Výjimečnost většiny karet nutila k nepřístojným řešením, takže bylo těžké odolávat, ale alespoň tak nevznikly nejhorší způsoby implementace, když už ne nejlepší.

dodgecity.DodgeCity
Rozšíření Dodge City mění omezení počtu hráčů, přidává jednu roli odpadlíka, nový druh stolních karet a značně rozšiřuje hrací karty a karty postav. Počet karet ve hře se tak zdvojnásobí s tím, že použití některých karet je složitější.

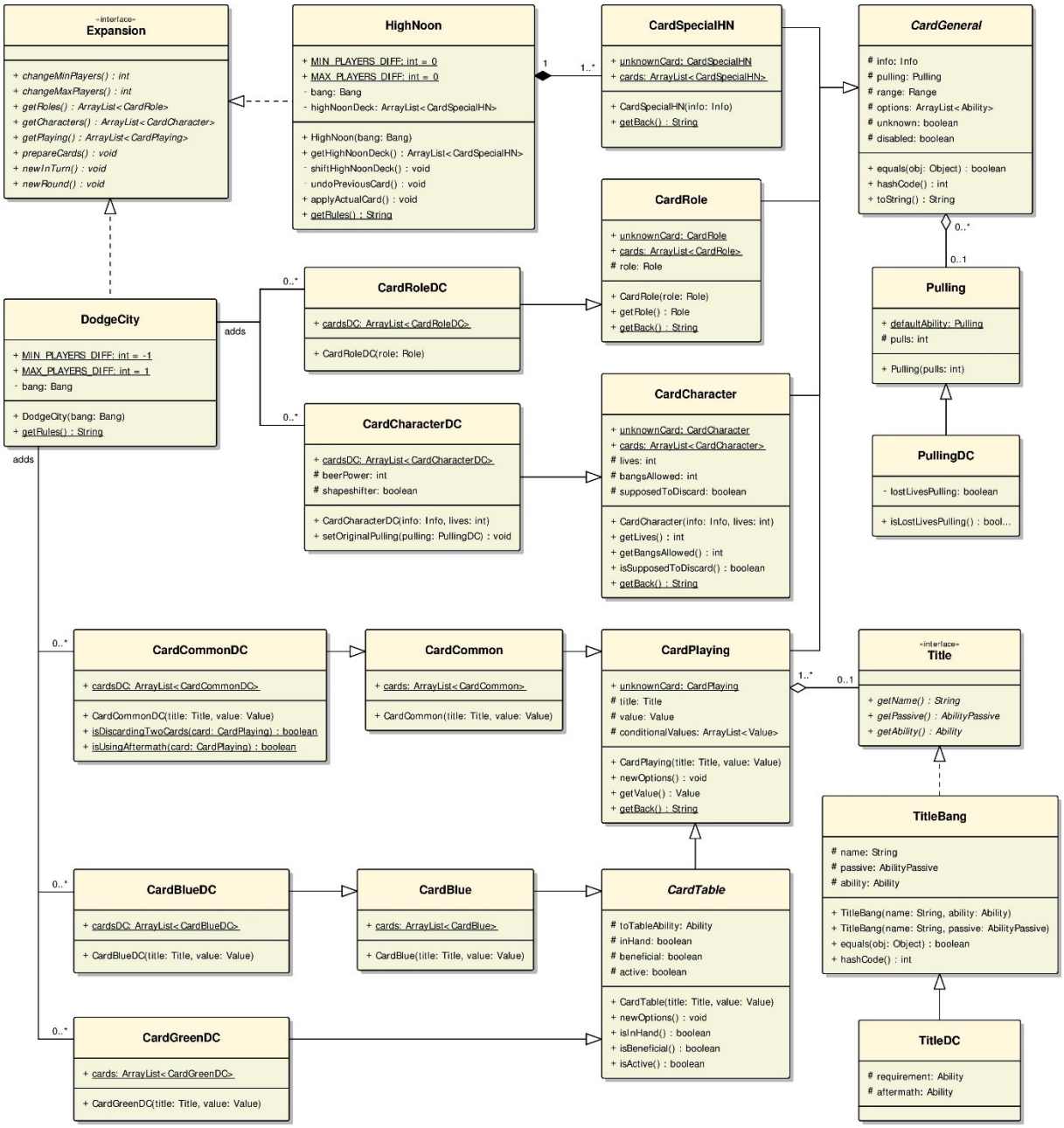
CardRoleDC, CardCharacterDC, CardCommonDC, CardBlueDC a CardGreenDC
Tyto třídy dědí ze tříd základního Bangu!, aby mohly obsahovat instance nových karet a zároveň doplňovat nové atributy.

dodgecity.TitleDC, dodgecity.PullingDC
Vytvoření těchto tříd dědících ze základní hry je rovněž ukázkou, jak logicky oddělovat nové vlastnosti karet patřící k rozšíření. Původní pojmenování všech tříd obou rozšíření bylo nevhodné (TitleDodgeCity, PullingDodgeCity atd.), pro přehlednost jsem proto zavedl zkratky jejich názvů.

highnoon.HighNoon
Rozšíření High Noon mění chod hry. Většinou se to týká vlastností hráčů a karet. Doplňování životů je celkem lehké, zajímavější je změna atributů jako je směr hry, počet líznutí, barva karet, blokace hracích karet a karet postav, počet dovolených Bangů! a oživení hráče na jedno kolo. Aktuálním řešením je zachovávat původní hodnoty spolu s danými atributy, lepší varianta by byla spíše uchovávat obě zdvojené hodnoty přímo ve třídách vlastností. Tak nebo tak, aby tyto atributy byly přidány jako část rozšíření, musely by se nejspíš zdědit a upravit téměř všechny základní třídy a to by se týkalo změny směru hry a oživení hráče i tříd hry a hráčů, což by jen kvůli dvěma proměnným typu boolean bylo spíše škoda. Ke změnám hry dochází začátkem kola, před změnou jsou všechny hodnoty nastavovány na původní.

CardSpecialHN
Jednoduchý druh karet rozšiřující CardGeneral. Nejsou ani tak důležité vlastnosti jako jejich efekt.

Obrázek 7 - Diagram rozšíření



3.2 Vztah server-klient

Zprvu jsem zkoušel tuto část programu takovým způsobem, že jeden z hráčů (tzv. Host) vytvořil hru a představoval roli serveru. Komunikace mezi ním a dalšími hráči probíhala pomocí jednoduchých řetězců, které se musely parsovat [3]. To bylo pro pár testů poučné, nicméně ani jedno by nebylo správným řešením. Bylo potřeba vytvořit samostatný serverový program, na kterém by probíhaly veškeré hry, a každý z klientů by k němu byl připojen. Stejně tak bylo potřeba nahradit jednoduché *DataInputStream* a *DataOutputStream* za object streamy, které hlavně v kombinaci s Command vzorem (typ Behavioral) znatelně práci usnadňují [4].

balíky

connection	Balík s třídami Server a Client pro komunikaci. Na serveru jsou vlákna tříd ServerThread a ServerThreadAI, které spolu s třídou Client implementují rozhraní User, které je používáno pro většinu příkazů. Obsahuje některé z příkazů logicky patřící k nastavení Userů.
connection.bang	Zde jsou příkazy týkající se hry. Za zmínku stojí především StartGame a UseCard, které se dají považovat za základ řadiče, čili kontrolu nad hrou.
connection.bang.highnoon	V balíku je jen příkaz pro změnu High Noon karty. Všechny příkazy změny nějaké karty by se daly sjednotit do příkazu jednoho, ten by musel mít více atributů, což by bylo negativní spíše pro zátěž serveru při mnohonásobně větším počtu hráčů. I tak věřím, že příkazy by měly obsahovat co nejjednodušší atributy a aby jich bylo co nejméně.
connection.room	Třída Room obsahuje základní informace o hře i další informace, proto je vhodná k posílání klientům. V balíku jsou i příkazy rozhraní CmdRoom aktualizující místnosti her (triviálnější než CmdUser nebo CmdBang).

3.2.1 Komunikace

Velikost zasílaných a přijímaných herních příkazů se při testování pohybovala mezi 1,7 - 2,4 KB, průměrně 2 KB. Vyřazením duplicitních atributů příkazů by se tato hodnota dala zmenšit. Triviálnější příkazy jako posílání textové zprávy měly jen okolo 100 B, po konci hry (13 kol) bylo klientem získáno 804 KB dat a odesláno pouze 42 KB. Výpočet hodnot nebyl optimálně přesný, přesto vytvořil základní představu toho, jak díky příkazům a pár celočíselným hodnotám může být přenos obsahově nenáročný.

connection.User

Rozhraní. Klienti třídy Client i vlákna klientů na serverové straně ServerThread implementují rozhraní User, které tak na obou stranách zajišťuje metody pro posílání/přijímání objektů, ukončení spojení s klientem, získání jedné místnosti nebo celého seznamu místností, reakce na nové zprávy a operace s typem klienta a jeho pozicí v místnosti/ hře. Na serverové straně se vytváří instance místnosti i hry a klient vlastní kopie těchto instancí. Jakékoli změny jsou klientům ihned zasílány. Metody mají názvy *receive*, *send*, *getRoom*, *getBang*, *setSlot*, *setType*, *newMessage*, *softEnd*, *kickEnd* atd.

connection.Server
Server je jako samostatné vlákno, aby mohl být za pomocí knihovny Swing spuštěn v okně. Vyčkává na připojení uživatelů. Obsahuje všechny hry a klienty, do interní třídy Game zabaluje instanci místnosti, Bangu!, sdružené vlákna klientů a umělou inteligenci.

connection.ServerThread
Vlákno je vytvořeno a spuštěno po přijmutí klienta serverem. Na základě socketu vytváří object streamy. Podle zaslaných informací v Relation dojde k vytvoření/aktualizaci místnosti. Vlákno pak stále přijímá klientovy požadavky a na základě proběhlých událostí zasílá příkazy.

connection.Client
Jakmile je klient prezentační vrstvou vytvořen, snaží se připojit k serveru. Na to má nastaven dvousekundový time-out. Pokud pokus nevyjde (server není spuštěn, chyba spojení), program to uživateli ohlásí. Důležité je po vytvoření socketu nejprve zavést ObjectOutputStream a až poté ObjectInputStream, prohození může vést ke značným problémům.

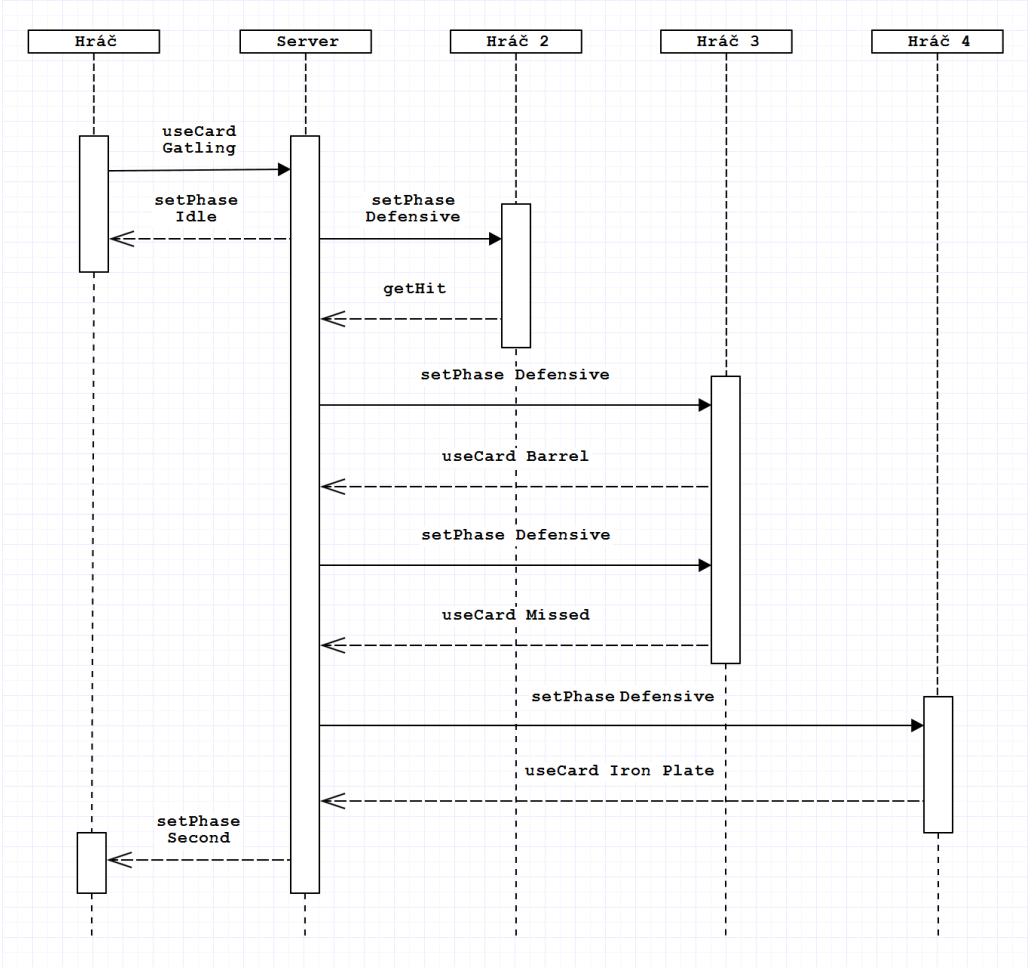
connection.Relation
Nastavuje vztah mezi serverem a klientem. Určuje typ klienta a případně ID místnosti, do které chce vstoupit.

rozhraní CmdUser a příkazy SetSlot, SetType, GetRooms, SendMessage, KickEnd, SoftEnd
Příkazy úzce souvisí s rozhraním User, logicky by nezapadaly k třídě místnosti ani Bangu! Například atribut *slot* pouze ukazuje na pozici hráče, kterého klient zastupuje, nijak to přímo neovlivňuje průběh hry.

rozhraní CmdBang a příkazy StartGame, UseCard, GetHit a další
Příkazy prováděné nad hrou, pro potřebu netriviálního rozesílání příkazů klientům je ale potřeba provádět *execute* metodu s parametrem User místo Bang. StartGame za předpokladu, že je vykonávána hostitelem a všichni hráči jsou připraveni, začíná hru. Podle nastavení místnosti se přidají rozšíření, nastaví počet a náhodné pořadí hráčů, neobsazené pozice se na straně serveru nahradí umělou inteligencí, karty se připraví a zamíchají. Ze serveru se pak rozešle počáteční stav hry, nechá klientům dost času na načtení obrázků a pošle příkaz začátku hraní. Na serveru jsou také nastaveny reakce na události - při smrti postavy se odhalí její role, při konci hry je vyhlášen vítěz a odhaleny všechny role hráčů a na začátku nového kola dochází k aktivaci karty High Noon.

Příkaz StartGame je vykonán jen jedenkrát, zatímco druhý nejdůležitější příkaz UseCard je pak vykonáván při většině akcí. Při jeho použití je určena abilita a potřebné pozice, kterých se abilita týká - zdrojová karta, cílový hráč nebo karta a třetí pozice se používá zřídka a to při Dodge City kartách, kdy je potřeba odhodit kartu vybranou dle vlastní volby. Na serveru pak dochází ke kontrole správnosti a při metodě *useAbility* k další řadě ověření podmínek. Pokud je karta použita, u některých karet (Dodge City) následuje další abilita, kterou provádí metoda *useAftermathAbility*.

Příkazem GetHit se hráč nechá v obranné fázi zasáhnout.



Obrázek 8 - Sekvenční diagram ukázky útoku na všechny ostatní hráče

Sekvenční diagram představuje použití karty *Kulomet* jako typickou ukázku interakce. Ve skutečnosti je při použití karty nastavena obranná fáze všem hráčům pod útokem najednou, dokud však je před některým hráčem jiný hráč na řadě s výběrem, musí čekat. Jakmile žádný z hráčů už není v obranné fázi, hráč na tahu pokračuje ve hře.

connection.room.Room
Důvod této třídy byl uveden při popisu balíku connection.room. Mám za to, že je tato třída opravdu potřebná. Provedení si však žádá úpravu k lepšímu.

rozhraní CmdRoom a příkazy SetName, SetExpansions, SetSlots, SetPlayer a SetReadiness
Oproti jiným příkazům se tyto týkají čistě místnosti a metoda *execute* vrací boolean. Pokud je true, automaticky se pošle klientům.

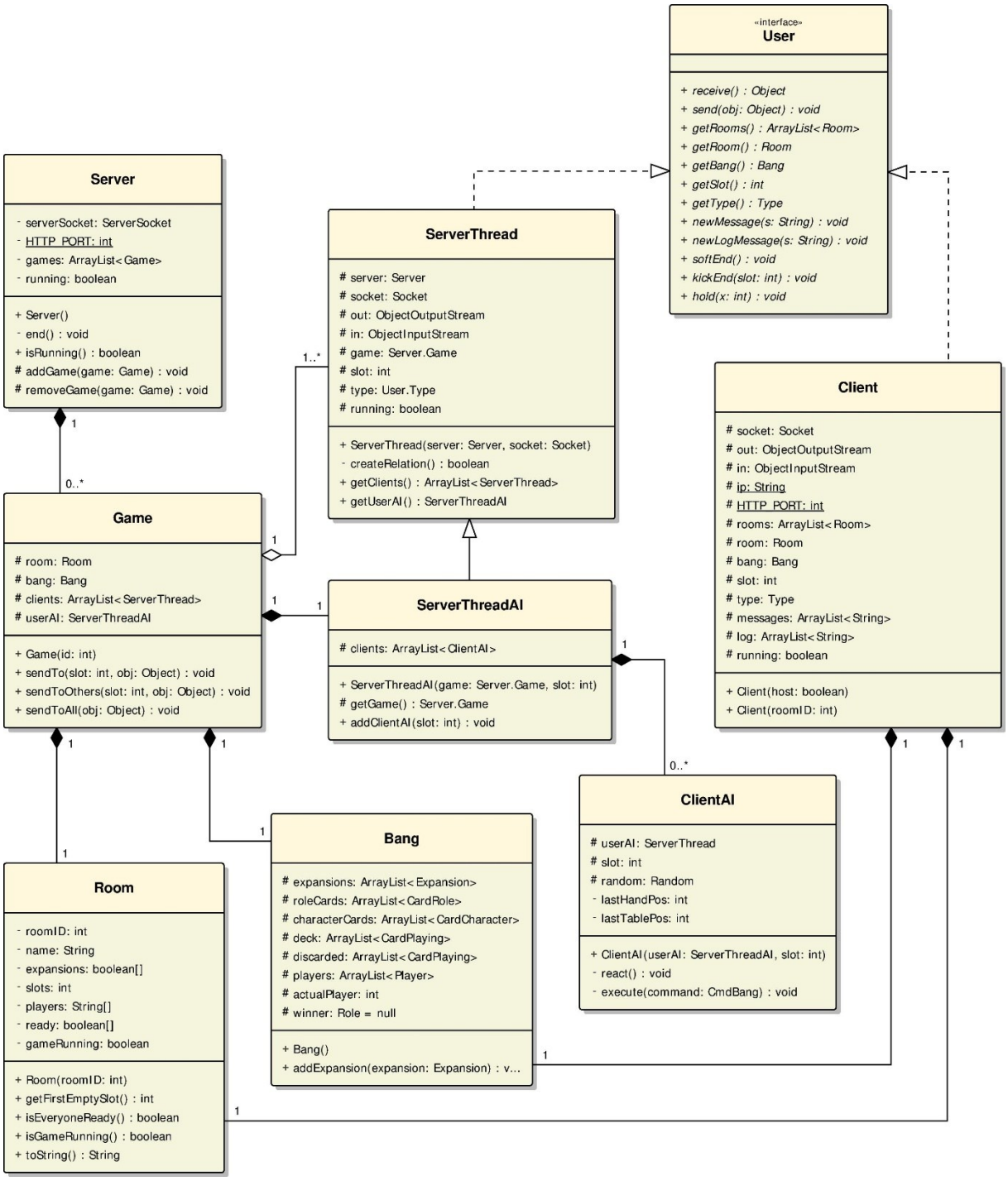
3.2.2 Umělá inteligence

K vytvoření AI byla nezbytná správná myšlenka. Proto jsem koncept začal vytvářet podle uvažování skutečných hráčů na začátku hry, kdy je známa pouze role šerifa a každý musí odhadnout, jaké role mají ostatní hráči, a na základě toho si vybrat, na koho útočit a komu pomáhat.

I když se zprvu zdála realizace nemožná, opravdu se dala vytvořit jednoduchá umělá inteligence, která se dokáže tvářit jako skutečný hráč. Hlavní nedostatek může nastat při situacích, kdy už jsou odhaleny všechny role stejného typu, a očekává se, že AI hráč znovu zváží jeho odhady.

connection.ServerThreadAI, connection.ClientAI
ServerThreadAI je vlákno pro imitaci skutečného klienta a obsahuje umělé hráče reagující na vyvolané změny - instance třídy ClientAI. Původně jsem používal reakci na změnu fáze, při které se spouštěly cykly, během kterých AI hráči prováděli své tahy. Toto však bylo pro hru spíše nebezpečné a navíc zasílání příkazů klientům blokovalo přijímání příkazů z jejich strany (zprávy chatu, opuštění hry). Proto jsem se vydal snazší cestou, kdy jsou AI hráči v časových intervalech vyzýváni k reakci na jejich situaci a pokud jsou v neaktivní fázi, reakce se jich netýká a nic neprovedou. Tím, že je ServerThreadAI spojen se hrou přímo na serveru, nepotřebuje příkazy přijímat ani zasílat, proto jsou metody *receive* a *send* prázdné.

Obrázek 9 - Diagram stran serveru a klienta



3.3 Prezentační část

Pro aplikaci je tato část programu neméně významná. Hra by měla vypadat moderně a přesto být nenáročná, aby nebyla ochuzena o hratelnost. Práce s alpha kanálem obrázků .png a všeobecně použití průhledných barev je pro Java komponenty celkem oříšek, ale námaha za to stála. Dosažení přijatelných výsledků spočívalo ve správném překreslování průhledných prvků i těch za nimi metodou *repaint*. Někdy jsem byl donucen použít až surový způsob a nebyl z toho nadšený, obzvlášť když se jedná o pouhou dvojrozměrnou grafiku.

balíky

program	Obsahuje .jnlp soubor a klientskou verzi programu.
program.server	Program ke spouštění instance serveru.
program.window	Veškerý obsah prezentační vrstvy. V balíku se nachází obrazovky menu a hry, třídy upravených JComponent, chat a loadery obrázků a zvuků.

související složky /images a /sounds

3.3.1 Program

Bylo potřebné v Netbeans spouštění přes Web Start správně nastavit [6] a aplikaci podepsat. Koupě důvěryhodného certifikátu pro malé aplikace je dle mého názoru naprosto nevýhodná. Na toto téma je spousta negativních ohlasů, protože uváděných 200-300 $ (cca 5 000,- Kč) za roční platnost rozhodně není zanedbatelných. Další možnost je používání vlastně podepsaného certifikátu, ke kterému budou mít uživatelé přístup. Jiná a nejspíše poslední možnost je přimět uživatele k nastavení výjimky pro aplikaci nebo ke snížení zabezpečení Javy. Nic z tohoto mi nepřijde jako správné řešení provozu hry.

program.LaunchBang
XML tagy .jnlp souboru definují název aplikace, vlastníka, popis, cestu k ikoně programu a další nastavení [5]. Nejdůležitější je určení metody *main* ke spouštění.

3.3.2 Pomocné třídy

program.GameImages
Při začátku hry načte všechny potřebné obrázky karet.

program.GameSounds
Při spuštění programu se načtou zvuky.

rozhraní MyComponent a třídy MyIcon, MyIconLabel, MyTextArea, MyTextField, MyTextLabel
Upravuje komponenty. Metody mají kratší názvy a podporují typy double, nastavují doporučené rozměry, třída MyIcon navíc mění velikost a úhel obrázků a třída MyIconLabel umožňuje průhlednost a záměnu na odstíny šedi [7]. Při práci s obrázky bylo podstatné zaměřit se na jejich kvalitu, protože

jejich čitelnost se netýkala pouze vizáže, ale i praktického využití [8]. Vyzkoušel jsem snad tři speciální knihovny pro kvalitnější zmenšení, ale čím větší úroveň kvality byla nastavena, tím byly spíše více rozmazané. Proto jsem se vrátil ke klasické bikubické interpolaci a raději zmenšil přímo obrázky v souborech, protože se to jevilo jako nejvhodnější řešení.

program.MyScrollChat
Jednoduchý chat složený z MyTextArea uvnitř JScrollPane, MyTextField a tlačítka na odesílání zpráv. Nastavením rozměrů se změní velikosti a pozice všech těchto komponent. Také umí zmenšit scrollbar.

program.Tools
Pomocná abstraktní třída se statickými metodami usnadňující práci s URL, soubory, obrázky a zvuky.

3.3.3 Obrazovky

Vzhledem hry a ovládáním se zabývá uživatelská dokumentace, více o jednotlivých sekcích je rozepsáno tam.

program.Screen
Abstraktní třída tvořící základ pro obrazovky menu a hry. Screen0 je menu, Screen1 až Screen5 jsou jednotlivé sekce.

program.ScreenGame
Obrazovka hry. Přistupuje ke klientské verzi Bang! hry a reaguje na její změny. Skládá se bočního panelu vytvořeného metodou *setSidePanel*, dále se vytvoří hrací plocha s obrázky karet, které jsou rozmístěny ve středu a po okruhu. Protože je ve hře celá škála karet a může ji hrát hodně hráčů, byla nutné co nejlépe využít prostoru obrazovky. Metoda *setSlotPosition* důvtipně vypočítá pozice hráčů podle jejich počtu a přehledně je rozmístí.

Nejdříve jsem zkoušel vypočítat body po obvodu obdélníku. Protože jsem po rozsáhlém pátrání nemohl najít žádný spolehlivý zdroj na toto konkrétní téma, musel jsem se držet příspěvku autora *belisarius* v jedné diskuzi na téma s názvem „Finding points on a rectangle at a given angle“. Ten poukázal na to, že podle úhlu *t* lze určit, kterou stranu obdélníku polopřímka od středu daným směrem protíná:

$$
\begin{aligned}
t \in \left(-\tan^{-1}\frac{b}{a}, \tan^{-1}\frac{b}{a}\right\rangle \quad &\rightarrow region\ 1\\
t \in \left(\tan^{-1}\frac{b}{a}, \pi - \tan^{-1}\frac{b}{a}\right\rangle \quad &\rightarrow region\ 2\\
t \in \left(\pi - \tan^{-1}\frac{b}{a}, \pi + \tan^{-1}\frac{b}{a}\right\rangle \quad &\rightarrow region\ 3\\
t \in \left(\pi + \tan^{-1}\frac{b}{a}, -\tan^{-1}\frac{b}{a}\right\rangle \quad &\rightarrow region\ 4
\end{aligned}
$$

Kde *a* a *b* jsou strany obdélníku, a podle toho se bod na základě středu $[x_0, y_0]$ vypočítá:

$$x = x_0 + \frac{a}{2}, y = y_0 + \frac{a}{2} \cdot \tan(t)\,, když\ je\ v\ regionu\ 1\ a\ 3$$
$$x = x_0 + \frac{b}{2 \cdot \tan(t)}, y = y_0 + \frac{b}{2}, když\ je\ v\ regionu\ 2\ a\ 4$$

Pozice hráčů sice byly rozmístěné, ale vypadaly nevyváženě. Proto jsem vyzkoušel výpočet bodů po obvodu elipsy uvnitř hrací plochy. Parametrické rovnice [9] vychází z jejího středu totožného se středem obdélníku a rozměrů *a* a *b*. Podle posunu (úhlu *t*) je pak bod na obvodu:

$$x = x_0 + a \cdot \cos(t)\,, y = y_0 + b \cdot \sin(t)\,, t\ \epsilon\ \langle 0, 2\pi)$$

Obvod prosté elipsy však nevyužíval celou plochu, jak bych si přál. Finálním řešením bylo vypočítat body na obvodech obdélníku i elipsy a jednoduše zprůměrovat obě varianty. S výsledkem jsem byl spokojen.

Velkou smyčkou se pak pro jednotlivé hráče nastaví role, postavy, jména, počty životů, jejich karty a zbytek obrázků. Zároveň jsou jim přidány reakce na průběh hry. Metody *addPlayerTargeting* a *addCardTargeting* nastavují výběr hráčů a karet, metoda *addDescriptionShowing* přidá zobrazení vlastností postav, metoda *createSymbols* přidává kartám obrázky jejich hodnot (můžou se měnit vlivem rozšíření High Noon), metoda *popup* ukazuje JPopupMenu s akcemi karet. Metody *startChoosing* a *stopChoosing* podle používaných abilit přepínají mezi způsoby vybírání.

3.4 Dodatek

U aplikace se předpokládá dobré využití paměti, ale také zabránění jejím únikům [10]. Proto jsem se snažil v rámci dostupného času co nejlépe promyslet vytváření nových objektů a práci s nimi. Bylo důležité si hlídat, co se s objekty děje. Pro kontrolu existuje řada programů, k základnímu ověření však stačilo užívanou paměť sledovat ve Windows Task Manageru. Je možné, že k mírným únikům dochází, zvláště při opouštění hry a zakládání nové. Navyšování paměti však bylo pozvolné, pro systém nijak nebezpečné. Při běžící hře bylo využití paměti spíš konstantní. Aplikace si i v tomto ohledu prošla zdokonalujícími změnami, další vylepšení by se mohlo týkat lepší kontroly nad vytvářenými JLabely s obrázky.

Hra měla být součástí herního portálu, avšak kvůli komplikacím byla programována nezávisle. I přesto podporuje základní potřebné nastavení a umožňuje používání vlastního chatu. Při vytváření by mi musela být známa struktura této společné aplikace a její rozhraní, podle kterých by se muselo držet. To by spíše ulehčilo veškerou práci, protože bych toho nemusel sám tolik implementovat. Díky pozdnímu představení aplikace portálu jsem tak alespoň získal představu, jak hodně by to ovlivnilo vývoj hry. Konstruovat části hry znovu by na druhou stranu bylo samozřejmě přítěží.

4. Uživatelská dokumentace

V této kapitole je vysvětlena aplikace z uživatelova pohledu, za pomocí snímků obrazovky a s náznakem tutoriálů se zde popisují její funkce. I když první dojem může pro některé hráče působit chaoticky, používání aplikace není složité díky snaze o co největší přehlednost - obrazovky mají za úkol jasně sdělovat, jak se v nabídce pohybovat a co se při hraní aktuálně děje.

Stejně jako se od většiny programů očekává, i tento usiluje o docílení co nejpředvídatelnějšího ovládání periferiemi. Možné nezřetelnosti jsou v následujících textech zmíněny a případně obhájeny nebo doplněny náznaky lepších řešení.

Dokud je aplikace ve stádiu, kdy nebyla vyzkoušena ani menším okruhem znalých hráčů, je jistý výskyt nedostatků. To však neznamená nic nečekaného.

4.1 O aplikaci a její požadavky

Obrázek 10 - Ikona hry

Název: Karetní hra Bang!
Velikost: 5,74 MB (5,76 MB s Java WebStart souborem a ikonou [OBRÁZEK 10])
Jazyk: český
Autor: Zdeněk Balicki (bal337)
Verze aplikace: 0.9

Hra je vytvořena především pro PC. Není nijak hardwarově náročná, požadavek rychlosti procesoru pro klientskou část je minimální, použití paměti se pohybuje okolo 150 MB, což v dnešní době není žádná překážka. Nejzákeřnější je načítání obrázků a zvuků ze souborů, to je ale vyřešeno načtením předem před začátkem samotného hraní. Načítání obvykle trvá pouze sekundu až dvě.

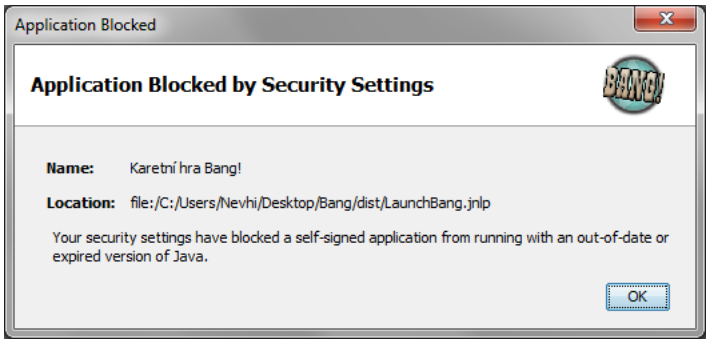
Protože se jedná o internetovou aplikaci, u které se počítá s častými aktualizacemi, její velikost nesmí být zanedbatelná. Z původních cca 60 MB .jar souboru byla za použití menších obrázků karet velikost hry zredukována 10×. Velké verze obrázků je možné si prohlédnout mimo hru.

4.2 Spouštění aplikace

Ke spouštění aplikace je potřeba mít na PC nainstalované JRE (Java Runtime Environment), nejlépe nejnovější, případně rovnou celé JDK JRE obsahující. Hra se sice dá vytvořit jako jeden .jar soubor,

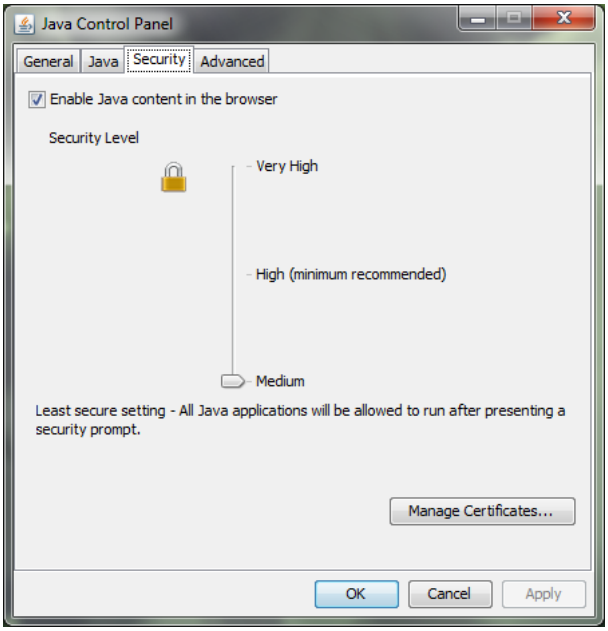
avšak mnohem zajímavější a výhodnější je využít Java Web Start možnosti. Při sestavení projektu se tak vytvoří velikostí malý .jnlp soubor, který obsahuje cestu k .jar souboru a při spouštění z webové stránky zajistí stáhnutí nejaktuálnější zveřejněné verze aplikace.

Protože je aplikace v tuto chvíli slabě podepsaná, může se jevit nedůvěryhodně - jak lze vidět na následujícím obrázku:



Obrázek 11 - Oznámení Application Blocked

Proto je potřeba nastavit nižší úroveň zabezpečení, což se ale doporučuje dělat pouze dočasně! Java Control Panel vypadá takto:



Obrázek 12 - Okno s nastavením zabezpečení Javy

Poté by aplikace měla být bez problémů spustitelná.

4.3 Uživatelské menu

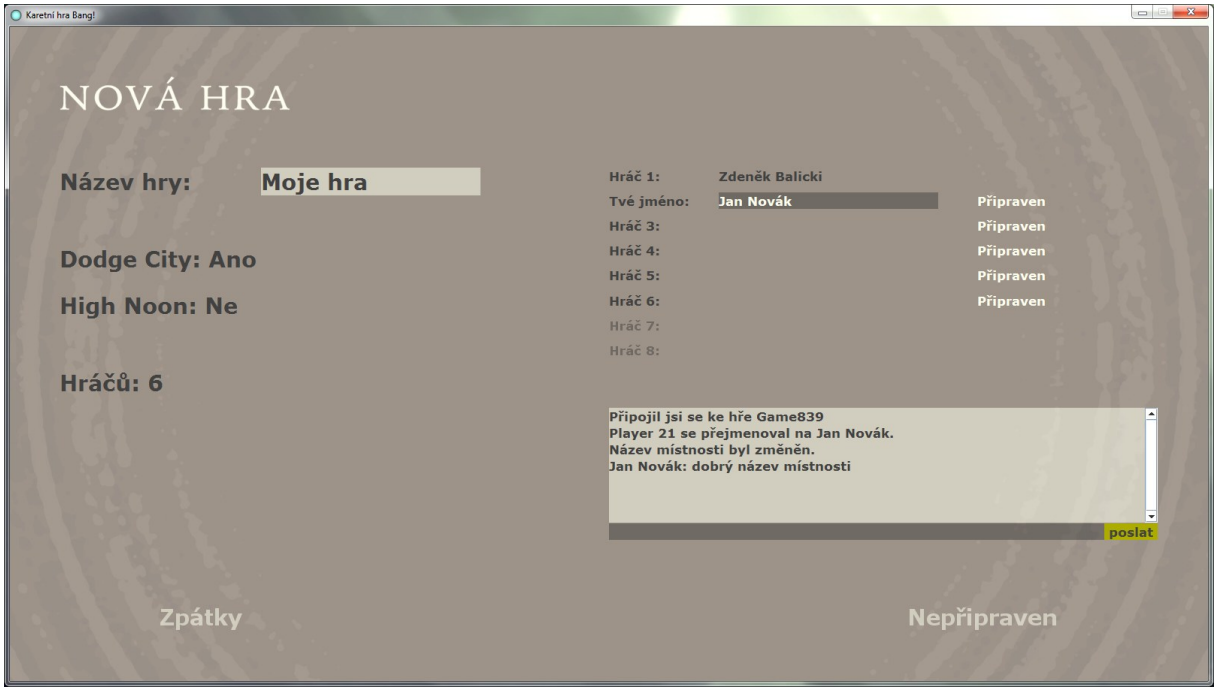
Menu je možné ovládat jak myší, tak klávesami. Při zapnutém zvuku je možné slyšet kliknutí na vybranou položku, pokud je zvuk nežádoucí, dá se vypnout v nastavení.

Obrázek 13 - Jednoduché, ale přehledné menu hry.

4.3.1 Nová hra

Pro vytvoření hry je aktuálně potřeba připojení k běžícímu serveru, i když má zakládající hráč zájem jen o hru jednoho hráče.

Obrázek 14 - Nová hra (verze hostitele)



Obrázek 15 - Nová hra (verze připojeného hráče)

Na předchozích snímcích ([Obrázek 14] a [Obrázek 15]) je vidět ukázka vytvoření nové hry. Hráč se přejmenoval na své jméno, připojil se nový hráč a také se přejmenoval. Hostitel místnosti nastavil název, nastavil rozšíření a změnil počet hráčů, další hráč mu to v chatu pochválil.

Všichni hráči v místnosti si tedy si smí změnit své jméno. Připojení hráči mají možnost změnit svůj stav z „připraven“ na „nepřipraven“, dokud nejsou všichni připraveni, hru nelze začít. Hostitel má možnost připojené hráče z místnosti vyhodit, v takovém případě se vyhozenému hráči objeví znovu obrazovka s menu a vyskočí okno s vysvětlením: „Byl jsi vyhozen z místnosti.“ Při hostitelově odchodu se novým hostitelem v místnosti stává hráč pod ním. Jestliže je místnost prázdná, zanikne.

Pokud nedojde při snaze vytvořit novou místnost ke spojení se serverem, obrazovka se načte, ale vyskočí okno s hláškou: „Nedošlo ke spojení se serverem.“

4.3.2 Připojení se ke hře

V této sekci se uživateli zobrazí název místnosti a nastavení her, při najetí myší na pole se ukáží přezdívky hráčů v ní. Aktualizace probíhá příkazem vpravo nahoře, případně může probíhat v intervalech. Kliknutím se do místnosti vstoupí.

Verze *spectatora* (diváka) není momentálně funkční, proto se vypisují jen místnosti neběžících her, ke kterým je možné se přidat. Pokud není založená žádná místnost, zobrazí se text: „Momentálně není založená žádná hra. “

Obrázek 16 - Výpis aktivních místností.

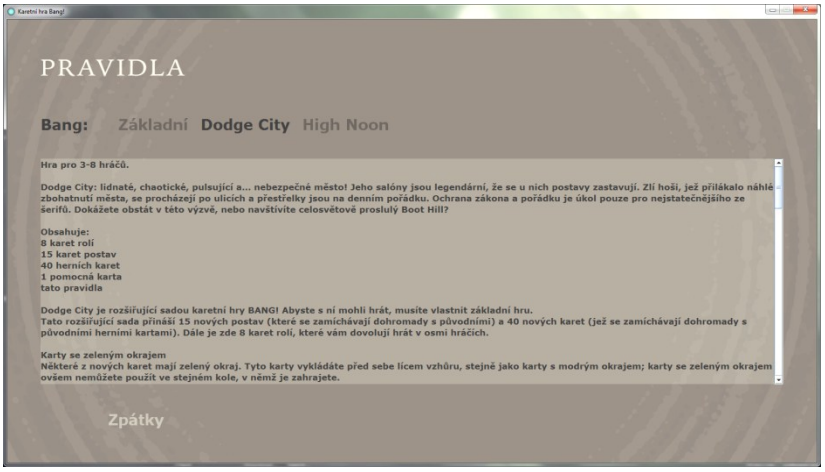
4.3.3 Další sekce

Menu obsahuje další jednoduché sekce, které v budoucnu mohou být více rozvinuty.

Nastavení hry:
Obsahuje především možnost výběru některého z výchozích rozlišení programu, ovlivňuje i velikosti načítaných obrázků karet. Zapínání zvuků se týká menu a základních herních zvuků a volba tlačítka je pro zobrazení možností nad kartami při hře. Dále je zde prostor například pro nastavení obtížnosti a rychlosti umělé inteligence.

Obrázek 17 - Sekce nastavení hry

Pravidla Bangu!:
Jednoduchá textová pravidla základní hry i obou rozšíření. Pravidel a FAQ nad nimi je spoustu, proto by bylo lepší využít například stránek www.bang.cz [1], ze kterých jsem hodně čerpal.



Obrázek 18 - Sekce pravidel

O aplikaci:
Informace nesoucí jména autorů, verzi programu a místo pro copyrighty nebo odkaz na webovou stránku.

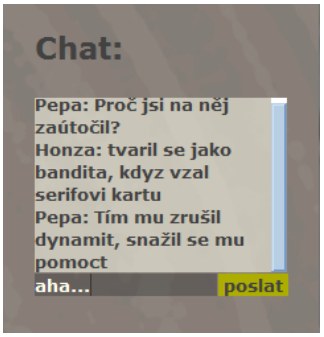
4.4 Herní obrazovka

Hra se ovládá myší, která je oproti zařízením bez ní velkou výhodou, jinak by ovládání bylo složitější. Obrazovka se skládá ze dvou jasně rozpoznatelných a přehledných částí, které jsou dále podrobněji popsány. Kvalitní interakce a zvýraznění děje jsou pro hraní důležité vlastnosti.

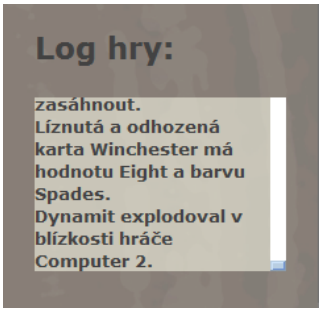
4.4.1 Boční panel

Panel slouží k využití širokoúhlosti dnešních monitorů. Nachází se na něm chat [Obrázek 19], log hry [Obrázek 20] a informace o stavu hry [Obrázek 21] - aktuální kolo, hráč na tahu a velikost karetních balíčků. V dalších verzích hry by se zde uplatnilo i nastavení hry, které je přístupné pouze ze základního menu. Při změnách rozlišení by se herní obrázky musely znovu načítat, jinak by jejich ostrost nebyla dostatečně kvalitní.

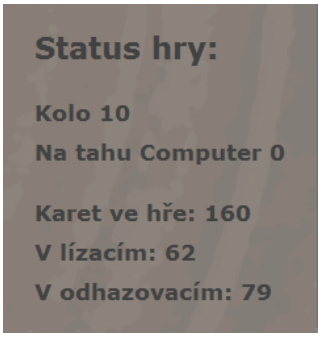
Protože je vnímání dění hry Bang! důležité i mimo hráčův tah, stávají se tyto prvky stěžejními pro potřebnou orientaci.



Obrázek 19 - Chat ve hře



Obrázek 20 - Log hry



Obrázek 21 - Info o stavu hry

4.4.2 Hrací plocha

Samotná hra pak probíhá na ploše, v jejímž středu se spolu s lízacím balíčkem nachází balíček odhazovací, který je na začátku hry prázdný. Po okruhu jsou ve skupinkách karty jednotlivých hráčů s tím, že své karty má hráč uprostřed dole. Nad kartami rolí a postav jsou jména hráčů a aktuální počet jejich životů, dole jdou vidět ve dvou řadách karty na ruce a karty na stole.

U balíčků se při hře zobrazují pokyny hráči. V některých případech se tam ukáže volba akce, která nemůže být vyvolána kartou (pustit ke hře dalšího hráče, nechat se zranit aj.). Akce nad kartami jsou během hráčova tahu vybírány standardně levým kliknutím myší na kartu. To znamená, že pokud si na začátku svého kola chceme líznout kartu z balíčku, klikneme na něj myší a vybereme možnost Líznout. V jiných fázích tahu se takto vybírá z možností konkrétních karet.

Obrázek 22 - Ukázka hry 1 (DC + HN)

Ukázka hry 1 [Obrázek 22] - hra osmi hráčů je v polovině prvního kola, směr hry je proti hodinovým ručičkám vlivem High Noon karty *Gold Rush*. Hráč je na konci svého tahu vyzván k odhození přebytečných karet, umělý hráč s postavou *Seana Malloryho* odhazovat karty nemusel, někteří hráči mají karet přebytek z důsledku použití karty *General Store* - každý hráč získal novou kartu do ruky.

Symbol bílé šipky vyznačuje hráče na tahu. Text u balíčků vyzývá ve třetí fázi vlastního tahu k odhození přebývajících karet. Při odhození poslední karty je pak automaticky na řadě další živý hráč.

Obrázek 23 - Ukázka hry 2 (základní Bang!)

Ukázka hry 2 [Obrázek 23] - základní hra bez rozšíření. Šerif na tahu si vybírá hráče, kterého chce uvěznit, při najetí myší na hráče se kolem něj objeví světlá aura povolující abilitu a při najetí myší na kartu jeho postavy se zobrazí i její vlastnost.

Při používání ability, která má regenerační účinky, je aura hráčů zelená. Pokud by zvolená akce z opodstatněného důvodu nešla provést (cíl není v dosahu, na šerifa nemůže být použito *Vězení* atp.), aura za skupinou karet by byla červená.

Toho jsi lze všimnout na následujícím případu:

Obrázek 24 - Ukázka hry 3 (HN)

Ukázka hry 3 [Obrázek 24]. Dokud před sebe bandita nevyloží z ruky silnější zbraň, šerif nebude kvůli vyložené kartě *Mustang* a vlastnosti postavy *Paula Regreta* v dosahu zbraně *Schofield* a při výběru bude znázorněn červenou aurou.

I při výběru karet (symbol *Vezmi kartu*, *Znič kartu* atd.) se stejně jako u výběru postav dá přehledně poznat, zda jsou vybírané karty mimo dosah, tím, že při najetí myší zůstanou nezvýrazněné. Toto je oproti hraní fyzické karetní hry velké a příjemné usnadnění, protože není potřeba vzdálenosti manuálně počítat a zároveň tak nedochází k lidskému faktoru pochybení.

Zvláštní případ vybírání karet je u většiny obyčejných herních karet rozšíření Dodge City, kdy se musí zvolit další karta z vlastní ruky, která bude spolu s použitou kartou odhozena. Při potřebě výběru se uprostřed hrací plochy objeví pokyn „Vyber svou kartu ke zničení.“ a následují další dvě možnosti - karta je spolu s druhou kartou použita, nebo se ještě musí vybrat cíl ability (hráč, případně něčí karta).

Pro chápání těchto pokynů se počítá se znalostmi pravidel, proto nejsou dále nijak přesněji rozepisovány. Logika interakcí je vytvořena tak, že pokud uživatel provede neplatnou operaci, nic se nestane nebo se na herní obrazovce zruší vybírání a operaci je možné provést znovu správně. I tak doporučuji přečtení pravidel.

Obrázek 25 - Ukázka hry 4 (DC + HN)

Ukázka hry 4 [Obrázek 25] - obranná fáze.

Hráči musí být ve střehu i mimo svůj tah. Když je někdo pod útokem, pozadí obrazovky se změní v barvy nebezpečí. Hráč je vyzván k obraně. Jestliže má před sebou vyložený *Barel*, měl by začít obranu s ním. Pokud šance k vyhnutí nevyjde, fáze zůstane stejná a má příležitost se dále bránit. V případě, že nemá kartu na odvracení útoku, musí kliknout na akci „Nechat se zasáhnout“ - dobře viditelnou uprostřed.

Hráči pod drnem se od živých rozlišují desaturací svých karet. To se dá postřehnout na předchozím obrázku. Jakmile je při smrti hráče splněna podmínka konce hry, objeví se okno s vyhlášením vítěze nebo strany více vítězů. Pokud je vítěz v případě hry více odpadlíků nejednoznačný, je k roli připsáno i jeho jméno. Při konci hry jsou samozřejmě odhaleny role všech hráčů. Na hrací ploše se objeví informující text a po odsouhlasení výsledku hry zůstane použitelný jen chat a tlačítko pro odchod ze hry.

Pro Bang! turnaje byl vytvořen bodovací systém, který teď hra neumožňuje. Časem zde tedy přibude i tabulka s body za hru. Dokonce by se dala zvážit i možnost hraní turnajů přes internet, protože takové online události jsou velice populární.

Obrázek 26 - Ukázka hry 5 (DC + HN)

Ukázka hry 5 [Obrázek 26] - pěkně proběhnuté utkání se 4 umělými hráči. Skutečnému hráči v roli pomocníka se na poslední chvíli nepodařilo ubránit šerifa a zvítězili oba bandité, i když jeden z nich je už mrtvý.

4.5 Umělá inteligence

Pokud není dostatek hráčů ke hře podle přání, nabízí se využití umělých hráčů. I když je doporučeno AI hráče používat spíše jen jako doplnění počtu, s umělou inteligencí se dá hrát i sólo hra. Umělí hráči mají oproti skutečným zobrazeny světlejší jména. Stejně jako při hře jen skutečných hráčů, i v tomto případě dochází k náhodnému promíchání pozic.

Umělá inteligence této základní obtížnosti se umí bránit, vyložit karty před stůl, doplnit si životy, pokud jim chybí, líznout karty navíc, ale dokonce i útočit na teoretické nepřátele, vybírat si za odpadlíka stranu mezi dobrem a zlem, dát někomu kartu *Vězení* nebo mu třeba zrušit stolní kartu. I když je tento způsob umělé inteligence triviální, dá se takto dosáhnout složitější úrovně hraní a to potom i klidně ve hře jednoho hráče. Už i tato úroveň se dá jistě podcenit a je schopna hrát lépe než někteří začátečníci.

Když se někdo rozhodne opustit hru, je nahrazen umělým hráčem, aby ostatní hráči mohli v hraní pokračovat.

4.6 Chystané změny

Aplikace má jistě spoustu nedostatků. Při jejím vytváření byla použita snaha „Co nefunguje, nechť dočasně funguje alespoň nějak“. To se týká převážně pár případů, kdy je možnost hráčova volby udělána automaticky náhodně za něj. S funkčním základem programu je však dopilování těchto nesrovnalostí mnohem snazší, než mít před sebou program, který umí perfektně těchto pár činností, ale zbytek funkcí nedělá nic.

Prvním takovým adeptem by mohla být například dříve uvedená karta *General Store* (*Hokynářství*), při jejímž použití se na stůl má vyložit z lízacího balíčku přesně tolik karet, kolik je ve hře živých hráčů. V pořadí od hráče na tahu si každý vybere jednu z nich a vezme do ruky. Použití karty bylo zjednodušeno tak, že každý živý hráč získá z lízacího balíčku kartu bez možností volby.

Koncept řešení výběru karet pěti hráčů by pak vypadal nějak podobně jako okno uprostřed hrací plochy:

Obrázek 27 - Koncept karty *General Store*

Je téměř jisté, že hráč, který použil *Hokynářství*, by si vzal kartu *Wells Fargo*, druhý na řadě dle libosti a třetí hráč by tak měl na výběr tři zvýrazněné karty. U již vybraných karet by bylo napsáno, kdo si jakou vzal. V dočasné verzi funkčnosti by tak první hráč na řadě bez možnosti výběru získal kartu *Punch* (*Úder*).

Rozšíření High Noon je například téměř plně funkční, stejně jako valná většina obyčejných, modrých i zelených karet základního Bangu! a Dodge City. Největšími nedostatky trpí pár karet se symbolem knížky a spousta karet postav nemá implementovány jejich vlastnosti, a to především z důvodu jejich množství (31 téměř různých postav). Tento problém dělá hru nevyváženou, proto by zaměření na tuto část hry bylo více stěžejní než na třeba výše zmíněnou kartu *Hokynářství*.

Mrtvé postavy momentálně nepřichází o své hrací karty - z důvodu přehlednějšího testování, stejně jako jsou zatím neplatné „Tresty a odměny“ za zabití hráče.

5. Závěr

Touto prací jsem předvedl své řešení online karetní hry Bang! v jazyce Java. Světu jsem nejspíše ničím novým nepřispěl, avšak tyto zkušenosti se pro mě staly obrovským přínosem. Při vytváření aplikace jsem pochopil mnoho mi dosud nejasných základů programování a vyzkoušel nové, složitější způsoby realizace cílených výsledků. I když mi zpočátku přišla Java oproti jiným jazykům cizí, můžu s jistotou říci, že tomu tak již není.

Kromě občasných problémů a překážek probíhalo programování se zápalem a rozhodnutím dosažení co nejzajímavěji a nejoriginálněji vypadající hry. Dle mého názoru by bylo efektivnější začít s prací už na začátku studia a postupně aplikaci rozvíjet. Nejtěžší situace totiž nastávaly při přidávání nových funkcí, kdy bylo nutné dosavadní kódy zpětně upravovat. Čím složitějším se program stává, tím více se stává náchylným a i malá změna je schopna jej globálně ovlivnit.

Konzultace mě bavily a pomáhaly mě vést správným směrem. Sám jsem také často zpozoroval vlastní chyby a snažil se přicházet na lepší varianty řešení, kdy bylo potřeba mít dostatečné úsilí ke změnění několikrát až poloviny kódů. A stejné tendence ke zlepšování vnímám i teď - ke konci bakalářské práce, který je však zároveň i začátkem další etapy. Zda se v praxi znovu setkám s Javou/Web Startem je ve hvězdách, ale i tak jsou samozřejmě všechny získané vědomosti velmi užitečné. Každým vypracovaným projektem poznávám nové návrhové vzory a jejich aplikací jim lépe rozumím.

Upřímně mohu oznámit, že bych ve vylepšování a rozšiřování aplikace rád pokračoval, ale spíše za předpokladu pomoci dalších osob, protože navzdory tomu, jak jednoduchá se hra Bang! zdá být, má celou řadu zajímavých a komplikovaných pravidel, které je o to problematičtější vnést do kódu. Časová konzumace byla od počátku kritická, představa o perfektním programu jednoho autora je trochu děsivá - obzvlášť při nutnosti vytváření originální grafiky karet a zvuků hry i přesto, že základní obrázky hry jsou vytvořeny mnou. Aby hraní bylo na úrovni oficiální, nedávno vydané a komerční verzi, musel by se i vymyslet způsob testování a hlášení chyb, popř. nedostatků verze mé. To by vyžadovalo další čas.

Použitá literatura

[1] ŘÁBEK, Stanislav. SRAB. *BANG.cz: Stránky o karetní hře Bang!* [online]. 2005 [cit. 2014-04-16]. Dostupné z: http://www.bang.cz/

[2] LEAHY, Paul. Using Java Naming Conventions. ABOUT. *About.com* [online]. 1999 [cit. 2014-04-16]. Dostupné z: http://java.about.com/od/javasyntax/a/nameconventions.htm

[3] TRAVIS, Greg. Building a Java chat server. IBM. *DeveloperWorks* [online]. 2001 [cit. 2014-04-16]. Dostupné z: http://www.eecs.yorku.ca/course_archive/2011-12/W/3214/j-chat-ltr.pdf

[4] GAMMA, Erich, Richard HELM, Ralph JOHNSON a John VLISSIDES. GANG OF FOUR. *Návrh programů pomocí vzorů: Stavební kameny objektově orientovaných programů*. 1. vyd. Praha: Grada, 2003. 386 s. ISBN 8024703025.

[5] JNLP File Syntax. ORACLE. *Oracle Documentation* [online]. 2011 [cit. 2014-04-16]. Dostupné z: http://docs.oracle.com/javase/6/docs/technotes/guides/javaws/developersguide/syntax.html

[6] MAHARJAN, Narayan G. How to create JNLP (Web Start Launcher) from NetBeans 6. MAHARJAN, Narayan Gopal. *Java and FX: A perfect java blog* [online]. 2009 [cit. 2014-04-16]. Dostupné z: http://blog.ngopal.com.np/2009/10/15/how-to-create-jnlpweb-start-launcher-from-netbeans-6/

[7] Images - ImageIcon. SWARTZ, Fred. *Java Programming Notes* [online]. 2007 [cit. 2014-04-16]. Dostupné z: http://www.leepoint.net/notes-java/GUI-lowlevel/graphics/45imageicon.html

[8] Controlling Rendering Quality. ORACLE. *Oracle Documentation* [online]. 1995 [cit. 2014-04-16]. Dostupné z: http://docs.oracle.com/javase/tutorial/2d/advanced/quality.html

[9] Parametric Equation of an Ellipse. PAGE, John. *Math Open Reference* [online]. 2009 [cit. 2014-04-16]. Dostupné z: http://www.mathopenref.com/coordparamellipse.html

[10] REITBAUER, Alois, Klaus ENZENHOFER, Andreas GRABNER, Michael KOPP, Stephen PIERZCHALA a Steve WILSON. COMPUWARE CORPORATION. *Java Enterprise Performance* [online]. Software + Support, 2011 [cit. 2014-04-16]. ISBN 3868020403. Dostupné z: http://javabook.compuware.com

Seznam příloh

Součástí bakalářské práce je CD. Adresářová struktura disku je následující:

/Bang	implementace online hry Bang! v projektu IDE Netbeans
/big images	v podsložkách Bang, Dodge City a High Noon se nachází velké verze upravených originálních obrázků karet ve formátu .png
/executable	spustitelné soubory verze serveru i klienta, pro .jnlp variantu je potřeba nastavit úroveň zabezpečení - viz uživatelská dokumentace (Kapitola 4.2)
/thesis	bakalářská práce ve formátu .pdf